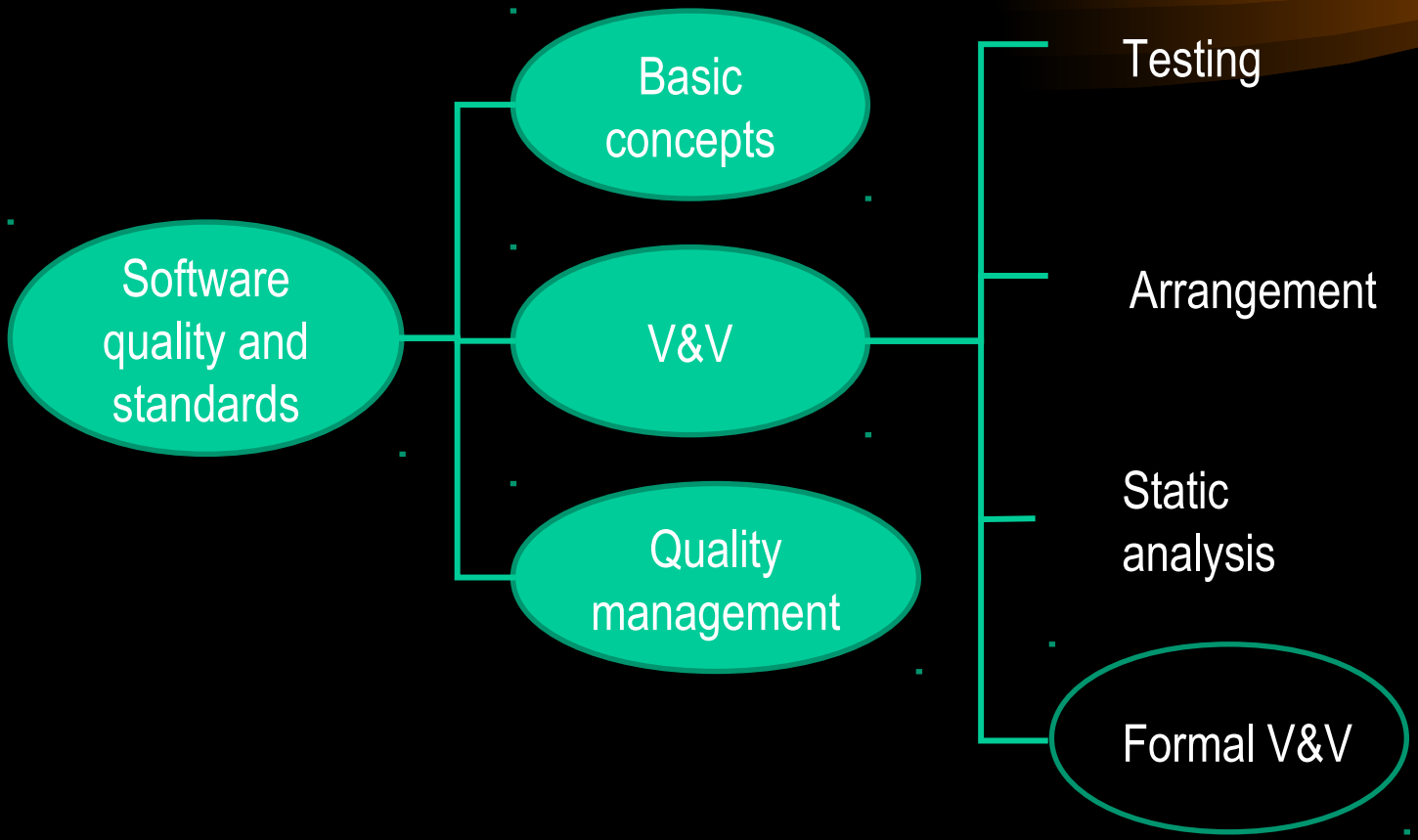
Methods for significant increase in software reliability

Jaak Tepandi
Tallinn University of Technology
Institute of Informatics
Moodle: „Software Quality (Tarkvara kvaliteet)”
Alternate download: tepandi.ee
Version 30.11.2016

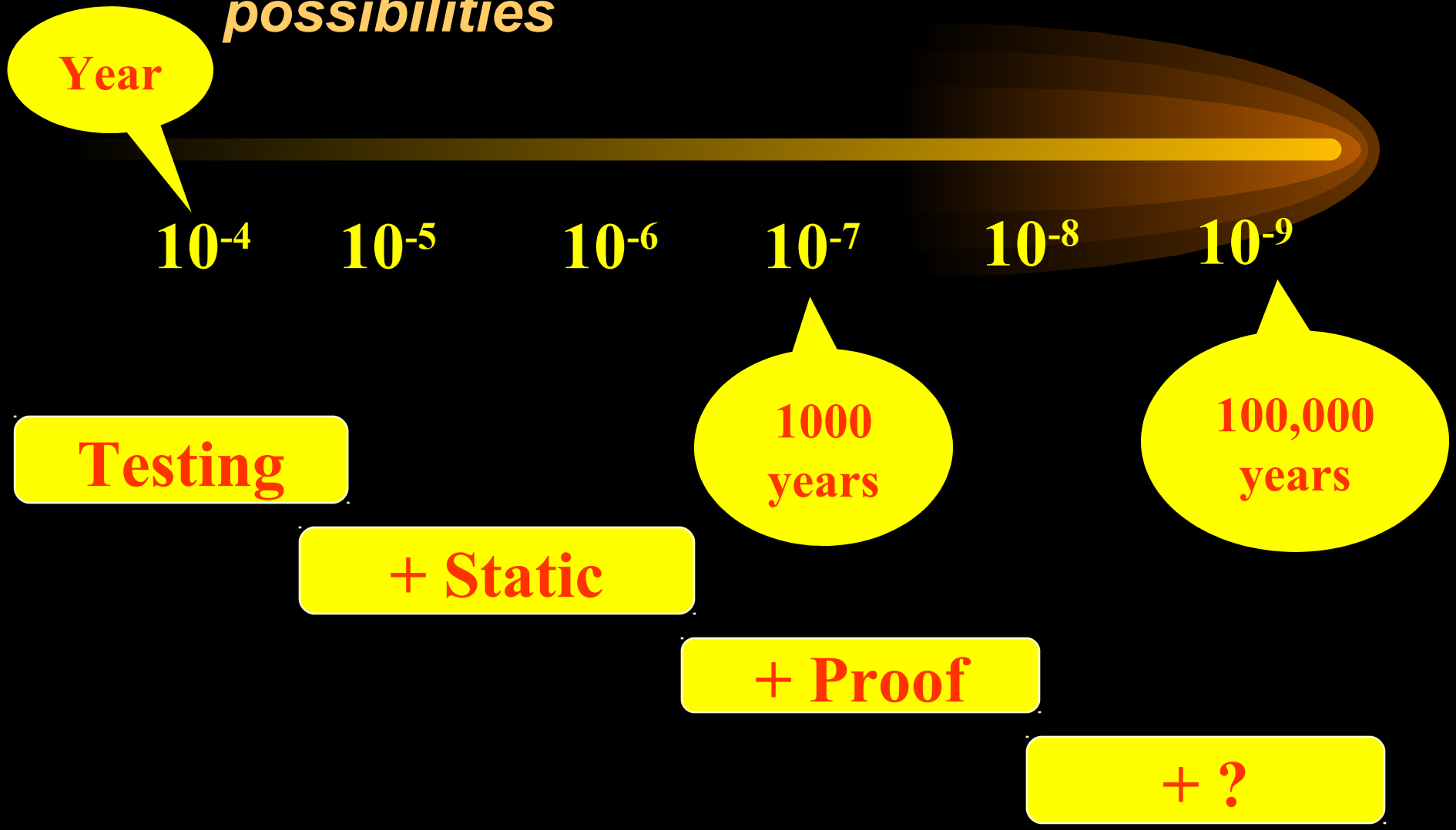
Context


Software quality and standards
Basic concepts
V&V
Quality management
Testing
Arrangement
Static analysis
Formal V&V

Topics

- Requirements
- N-version programming
- Fault tree analysis
- Formal verification
- Cleanroom software engineering
- Applications: Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408, “Common Criteria”)
- How solved for hardware?

Failures per hour: Requirements and possibilities*



*Extremely Improbable failure conditions are those having probability on the order of 1*10E-9 or less (for each flight hour) - Federal Aviation Adminstration

Jaak Tepandi

Summary of the methods

- concept
- preconditions
- advantages
- disadvantages
- results
- relationship to other methods
- evaluation
- tools

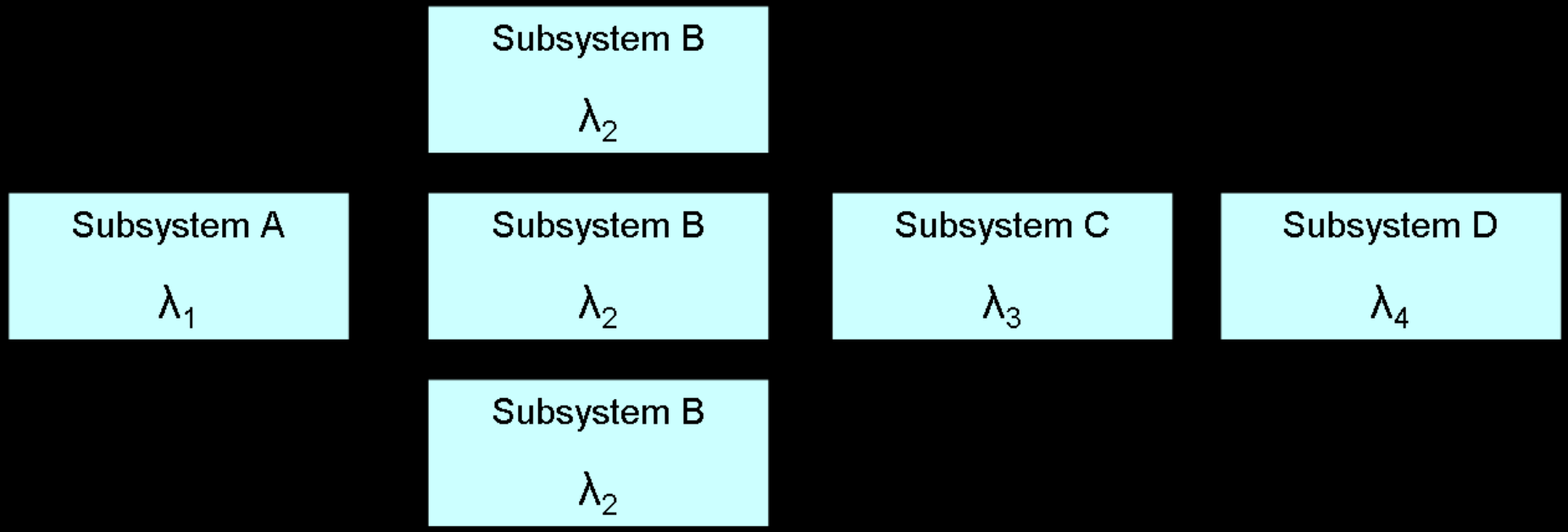
- concept
- preconditions
- advantages
- disadvantages
- results
- relationship to other methods
- evaluation
- tools

N-version programming

- Built in redundancy
- Multiple functionally equivalent programs
- Results compared, majority vote
- Good results for hardware
  - as high reliability as needed
- For software
  - Errors in spec
  - Similar human logic /errors in development

Redundancy in hardware

Each of the subsystems may be (recursively) redundant =>
it is possible to achieve high reliability



https://commons.wikimedia.org/wiki/File:Reliability_block_diagram.png

- concept
- preconditions
- advantages
- disadvantages
- results
- relationship to other methods
- evaluation
- tools

Fault tree analysis (1)

- Define the undesired outcome
- Understand the system
- Build the fault tree

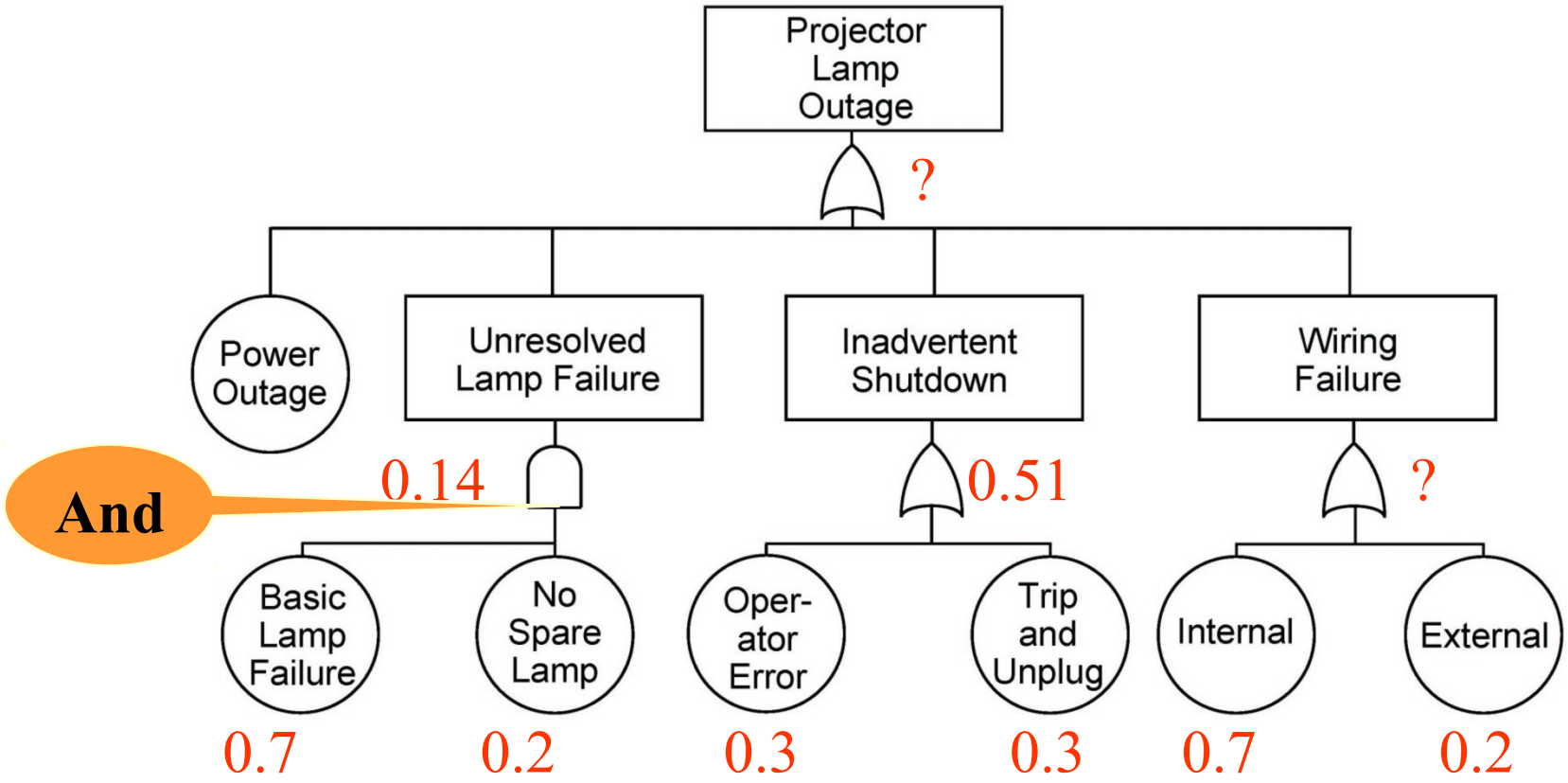
  - And-Or tree
  - The undesired outcome is the root
  - And-Or branches
  - Probabilities assigned to the causing effects (leaves)

- concept
- preconditions
- advantages
- disadvantages
- results
- relationship to other methods
- evaluation
- tools

Fault tree analysis (2)

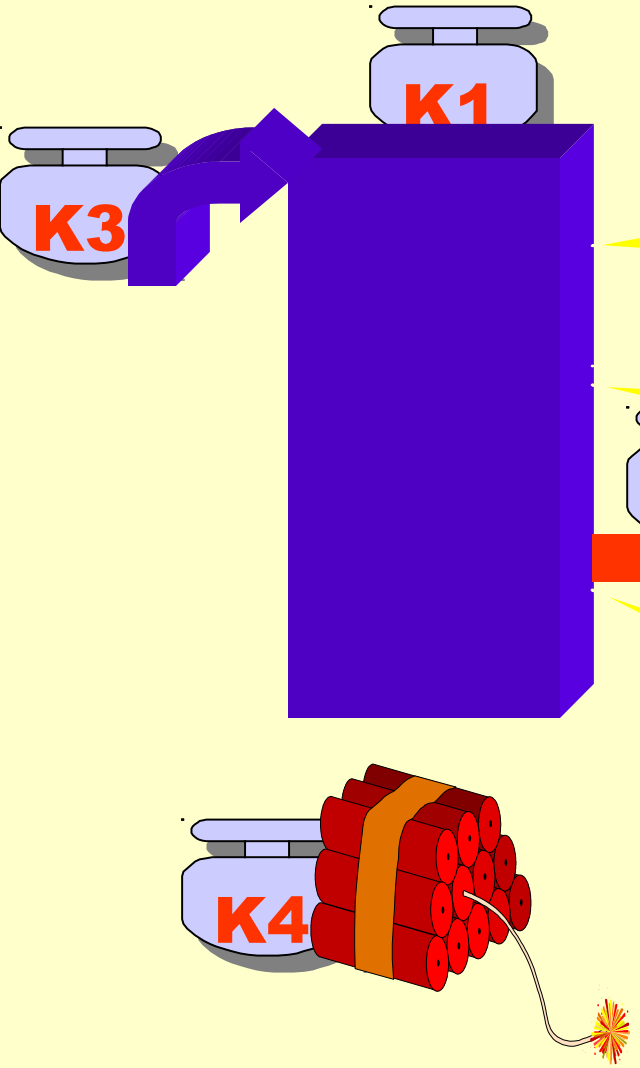
- Evaluate the fault tree
  - Is the failure probability acceptable?
  - What contributes most to the failure?
  - What could be improved to reduce the failure probability?
  - How could resources be optimized?
  - How should the system be designed?
  - How to create diagnostic processes?

Fault Tree Example



JACOBS SVERDRUP

Fault tree?



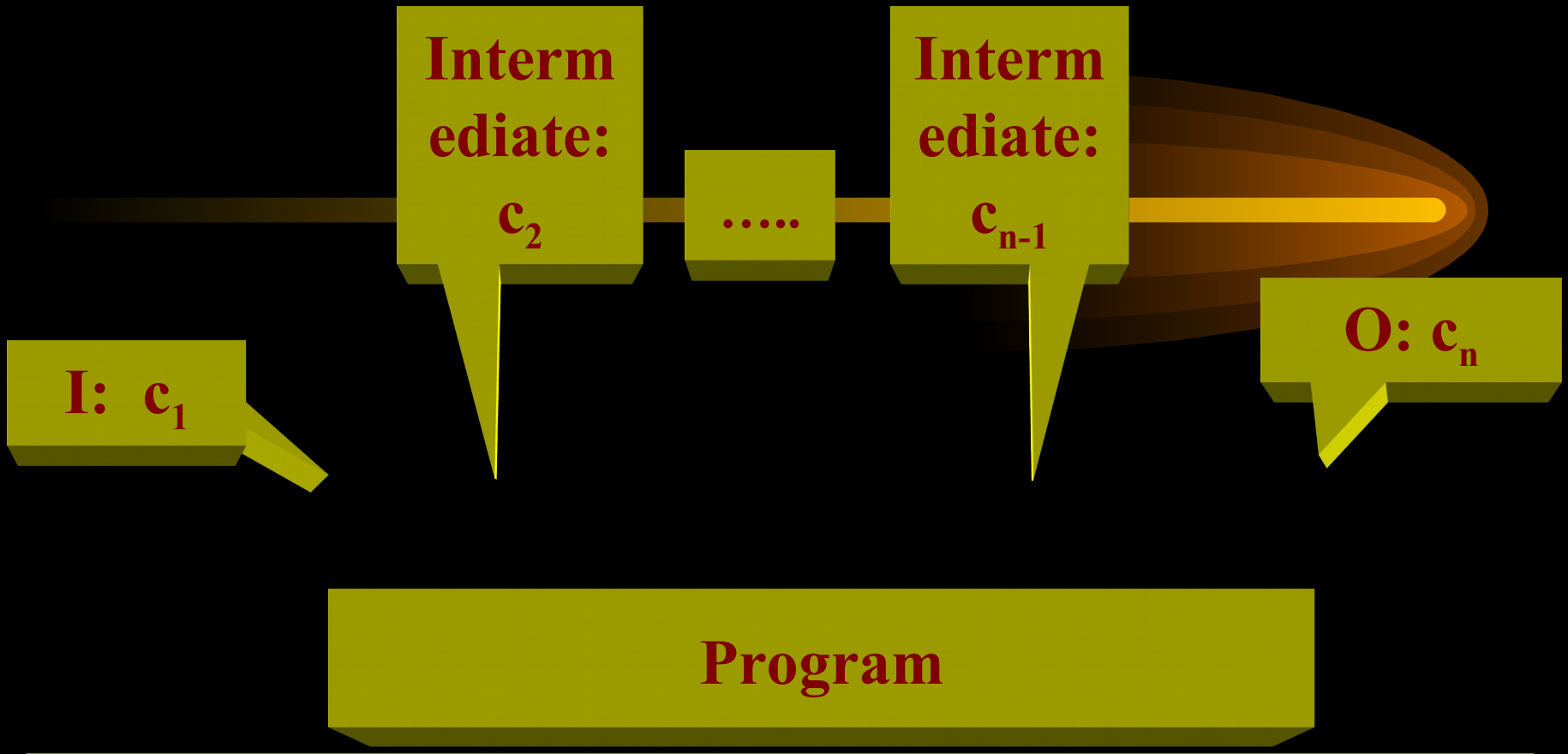
Water level here and above:
Heating=“Start”
Water = “Close”

Water level here and below:
Water = “Add”

Water level here and below:
Heating=“Stop”

Formal verification

- Need
- History
- Example
- How to make easier
- Levels
- Results



Mathematical models / formal representations:

- **For specifications (I/O, intermediate states)**
- **For the program**
- **For the proof**

Example: prove that this fragment …(does what?)

- A:= A+B;
- B:= A–B;
- A:= A–B

http://www.cs.uiowa.edu/~fleck/181content/

Specification + program

- {A=1 ∧ B=2}
- A:= A+B;
- B:= A–B;
- A:= A–B
- {A=2 ∧ B=1}

- http://www.cs.uiowa.edu/~fleck/181content/segproof.pdf

Making it easier

- Kernel + environment
- Kernel: prove that bad things do not happen... not necessarily that good things happen
- Environment is not so critical

Software/proof levels => proof problems

Software/proof levels:

- Specification: no previous level
- Design: if specification was formal
  - or for special properties (contradiction, redundancy etc)
- Source code: difficult but possible
- Proof methods: one-time effort
- Object code: prove the compiler, one-time effort
- Executable: prove the microcode, one-time effort
- Production/usage environment: different cases, usually cannot be proved

More problems

- Measurement units
- Programming technology, libraries etc
- Environment
- Operation
- …..

Verification results

- 10^{-7} … 10^{-8} failures per hour
- Not always sufficient
- Better than testing

Cleanroom software engineering

- Development and certification of high-reliability software systems under statistical quality control
- Name from hardware Cleanrooms
- Focus on defect prevention rather than defect removal
- Combines mathematically based methods of software specification, design, and correctness verification with statistical, usage-based testing to certify software fitness for use
- Cleanroom Software Engineering Reference Model - a set of 14 Cleanroom processes and 20 work products
- Cleanroom process flow
  http://www.sei.cmu.edu/library/abstracts/reports/96tr022.cfm p 26

Common criteria (ISO/IEC 15408 Evaluation criteria for IT security)

- An example of applying formal methods
- Seven assurance levels (part 3 Ch 7)
- Formal methods from level 5
- Target of Evaluation (TOE) — An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation (part 1)
- Number of certified products: 2191 (active, Nov 2016)

http://www.commoncriteriaportal.org/products_STAT.html

http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

How solved for hardware?

- Redundancy
- Simpler functions
- High reliability components

Popular

D. E. Knuth: "Beware of bugs in the above code; I have only proved it correct, not tried it." (http://www-cs-faculty.stanford.edu/~uno/faq.html).

Summary

- Requirements
- N-version programming
- Fault tree analysis
- Formal verification
- Cleanroom software engineering
- Applications: Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408, “Common Criteria”)
- How solved for hardware?

Additional reading (examples)

N-version programming,
http://www.cse.cuhk.edu.hk/~lyu/book/sft/pdf/chap2.pdf

Fault tree analysis https://en.wikipedia.org/wiki/Fault_tree_analysis

Program verification
https://en.wikipedia.org/wiki/Formal_verification#Software
Program proof example:
http://homepage.cs.uiowa.edu/~fleck/181content/seqproof.pdf,
http://homepage.cs.uiowa.edu/~fleck/181content/assignax.pdf,
http://homepage.cs.uiowa.edu/~fleck/181content/assignax.pdf

Cleanroom Software Engineering
ftp://ftp.sei.cmu.edu/pub/documents/96.reports/pdf/tr022.96.pdf

ISO/IEC 15408. Evaluation Criteria for IT security
http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html